LEC15: HEURISTIC SEARCH (247-)
Backtracking gave us a method to find the best of all possible solutions, as scored by a given objective function. However, any algorithm searching all configurations is doomed to be impossible on large instances.

Heuristic methods provide an alternate way to approach difficult combinatorial optimization problems.

We look at three different heuristic search methods:

- Random sampling,
- gradient-descent search, and
- simulated annealing.
TWO COMMON COMPONENTS

- **Solution space representation:** complete yet concise description of the set of possible solutions for the problem.
  - TSP: solution space consists of \((n-1)!\) elements - all possible circular permutations of the vertices, represented using an array S of \(n - 1\) vertices.

- **Cost function:** cost or evaluation function to access the quality of each element of the solution space.
  - TSP: given candidate solution S, cost function is just the sum of the costs involved.
RANDOM SAMPLING  AKA MONTE CARLO METHOD

- Repeatedly construct random solutions and evaluate them, stopping as soon as we get a good enough solution, or (more likely) when we are tired of waiting.
- Report the best solution found over the course of our sampling.
WHEN MIGHT RANDOM SAMPLING DO WELL?

- When there are a high proportion of acceptable solutions
  - Ex> Finding prime numbers
- When there is no coherence in the solution space - there is no sense of when we are getting closer to a solution
  - Ex>Finding who of your friends has a social security number that ends in 00.
LOCAL SEARCH

- A local search heuristic starts from an arbitrary element of the solution space, and then scans the neighborhood looking for a favorable transition to take.
  - EX: TSP, this would be *transition*, which lowers the cost of the tour.

- We want a general transition mechanism that takes us to the next solution by slightly modifying the current one.
  - Ex: TSP: swap the current tour positions of a random pair of vertices *Si* and *Sj*.

![Image of TSP tours before and after swapping vertices 2 and 6.]
WHEN DOES LOCAL SEARCH DO WELL?

- When there is great coherence in the solution space
  - Cost function is either convex or concave

- Whenever the cost of incremental evaluation is much cheaper than global evaluation
  - \textit{EX > TSP:} It costs $\Theta(n)$ to evaluate the cost of an arbitrary $n$-vertex candidate TSP solution. Once that is found, however, the cost of the tour after swapping a given pair of vertices can be determined in constant time.
Simulated annealing is a heuristic search procedure that allows occasional transitions leading to more expensive (and hence inferior) solutions – which helps keep our search from getting stuck in local optima.

Simulated annealing comes from the physical process of cooling materials down to the solid state.

A particle’s energy state jumps about randomly, with such transitions governed by the temperature of the system where Transition probability $P(e_i, e_j, T)$ from energy $e_i$ to $e_j$ at temperature $T$:

$$P(e_i, e_j, T) = e^{\frac{e_i-e_j}{k_bT}}$$
Simulated-Annealing()
Create initial solution $S$
Initialize temperature $t$
repeat
  for $i = 1$ to iteration-length do
    Generate a random transition from $S$ to $S_i$
    If $(C(S) \geq C(S_i))$ then $S = S_i$
    else if $(e^{(C(S) - C(S_i))/(k \cdot t)} > \text{random}[0, 1])$ then $S = S_i$
  Reduce temperature $t$
until (no change in $C(S)$)
Return $S$